

# Migrating Legacy Assets to Service Oriented Architecture (SOA): a DoD Example

Dennis Smith and Grace A. Lewis  
Integration of Software-Intensive Systems  
(ISIS) Initiative



**Software Engineering Institute**

**CarnegieMellon**

© 2007 Carnegie Mellon University

# Agenda

---

## SOA Basics ←

SMART (Service Migration and Reuse Technique)

Conclusions



# What is SOA?

---

Service-oriented architecture is a way of designing systems that enables

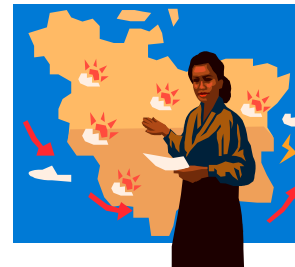
- Cost-efficiency
- Agility
- Adaptability
- Leverage of legacy investments



# Services

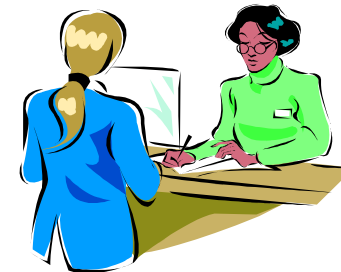
Services are reusable components that represent business tasks.

- Customer lookup
- Account lookup
- Credit card validation
- Weather
- Hotel reservation

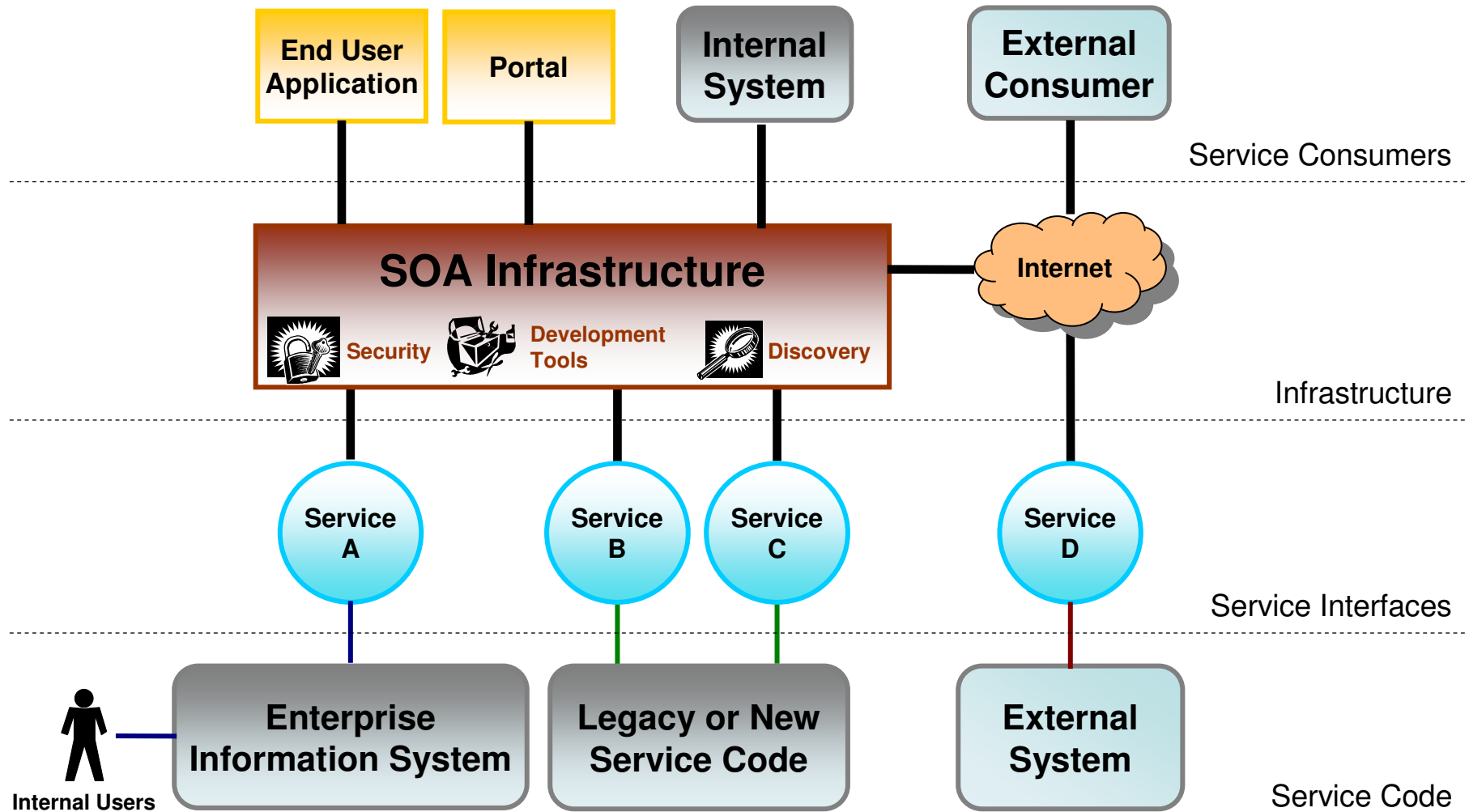


Services can be

- Globally distributed across organizations
- Reconfigured into new business processes



# Components of an SOA-Based System



# Agenda

---

SOA Basics

**SMART (Service Migration and Reuse Technique)** ←

Conclusions



# Reuse Challenges in SOA Environments <sub>1</sub>

---

Reuse at the service level is more complex than reuse at the module or component level

- From the service provider perspective
  - Bigger stakeholder community because services are typically reused at organization and sub-organization level
  - Services need to be as generic as possible so that they are of interest to multiple service consumers and at the same time need to add value to potential consumers
- From the service consumer perspective
  - Larger granularity may lead to larger incompatibilities



# Reuse Challenges in SOA Environments <sub>2</sub>

---

It may not always be possible to reuse functionality of legacy systems by exposing them as services

- Technical constraints due to the nature of the legacy system
  - A batch system needs to be exposed as a service for an interactive online web application
- Immature technology or lack of technology for a particular legacy environment

Cost of exposing a legacy system as services could be higher than actually replacing it with a new SOA-based system





# Bottom Line

---

There are issues to take into consideration that go beyond adding a service interface to an existing system

SMART is an initial approach to the identification and analysis of issues in migration to services



# SMART: Service Migration and Reuse Technique




---

SMART analyzes the viability of reusing legacy components as the basis for services by answering these questions:

- Does it make sense to migrate the legacy system to services?
- What services make sense to develop?
- What components can be mined to derive these services?
- What changes are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?

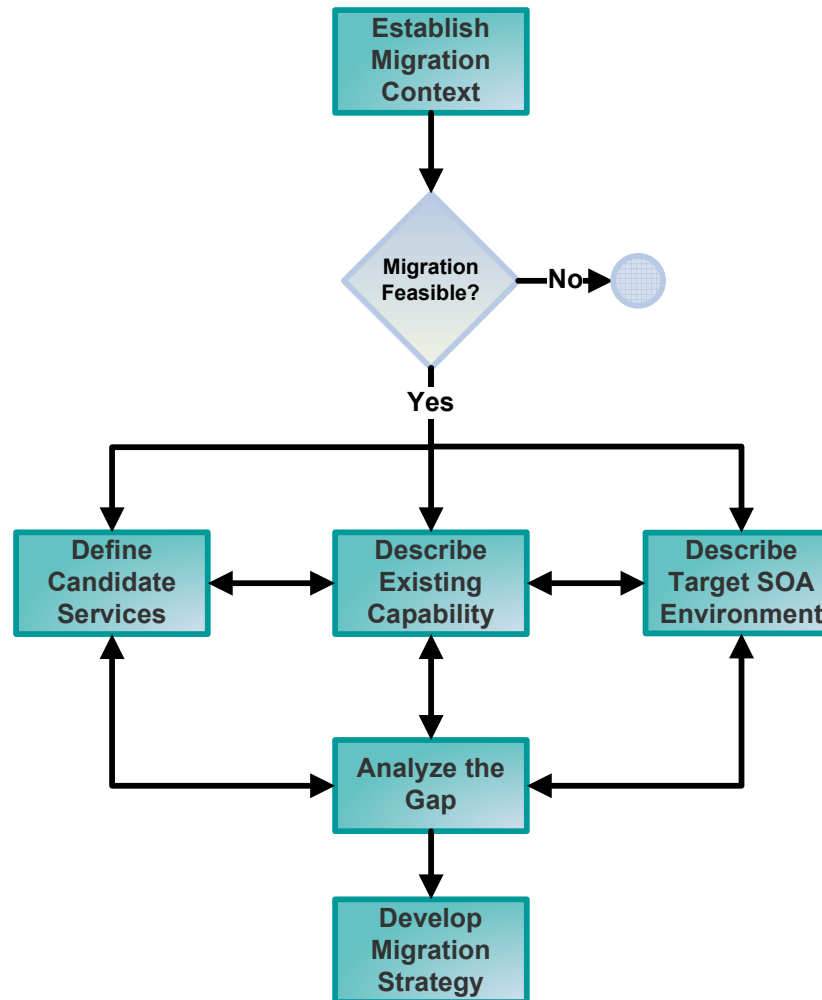


# Three Elements of SMART

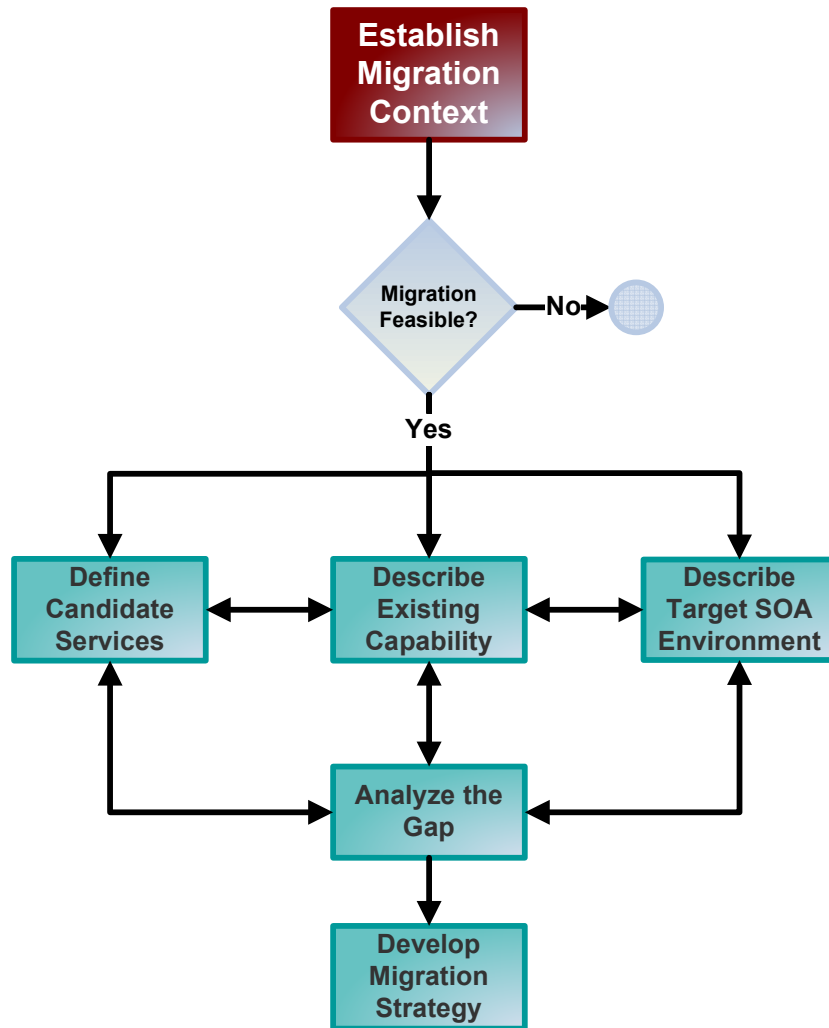
<b>Process</b> 	<b>Service Migration Interview Guide (SMIG)</b> 	<b>Artifacts</b> 
<p>Gathers information about</p> <ul style="list-style-type: none"> <li>• Goals and expectations of migration effort</li> <li>• Candidate services</li> <li>• Legacy components</li> <li>• Target SOA environment</li> </ul> <p>Analyzes gap between legacy and target state</p>	<p>Guides discussions in initial SMART activities</p>	<ul style="list-style-type: none"> <li>• Stakeholder List</li> <li>• Characteristics List</li> <li>• Migration Issues List</li> <li>• Business Process-Service Mapping</li> <li>• Service Table</li> <li>• Component Table</li> <li>• Notional SOA-Based System Architecture</li> <li>• Service-Component Alternatives</li> <li>• Migration Strategy</li> </ul>



# SMART Process Activities



# Establish Migration Context



Understand the business and technical context for migration

- Rationale, goals and expectations
- Technical and business drivers
- Programmatic constraints (e.g. schedule, budget)
- Previous related efforts or analyses

Identify stakeholders

- Who is driving and paying for the effort
- Who knows what about the legacy system and the target SOA environment
- Demand or need for potential services

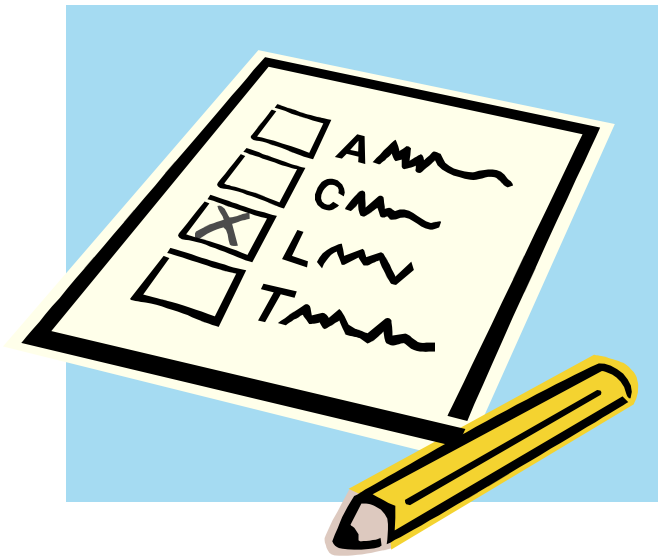
Understand legacy system and target SOA environment at a high level

Identify a set of candidate services for migration



# Identify a Set of Candidate Services for Migration

- Identify business goals
- Identify key business processes or mission threads that support these goals
- Identify common steps/tasks in these processes or threads
- Identify functionality from the legacy system to support these steps/tasks
- Negotiate! (Top-down and bottom-up approach)
- Select a number of the steps as candidate services



# Establish Migration Context: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
Goal and Expectations of Migration Effort	<ul style="list-style-type: none"><li>• What are the business and technical drivers for the migration effort?</li><li>• What are the short-term and long-term goals?</li></ul>	<ul style="list-style-type: none"><li>• No SOA strategy</li><li>• Goals for migration are not clear</li></ul>
High-Level Understanding of Legacy System	<ul style="list-style-type: none"><li>• What is the main functionality provided by the legacy system?</li><li>• What is the high-level architecture of the system?</li><li>• What is the current user interface to the system?</li></ul>	<ul style="list-style-type: none"><li>• Legacy system knowledge is not available</li><li>• Architectural mismatch</li><li>• User interface complexity hard to replicate in service consumers</li></ul>
High-Level Understanding of Target SOA Environment	<ul style="list-style-type: none"><li>• What are the main components in the target SOA environment?</li><li>• Is this the organization's first attempt to deploy services in this environment?</li></ul>	<ul style="list-style-type: none"><li>• Target SOA environment has not been identified</li><li>• No in-house knowledge of target SOA environment</li></ul>
Potential Service Consumers	<ul style="list-style-type: none"><li>• Who are the potential service consumers?</li></ul>	<ul style="list-style-type: none"><li>• Consumers for services have not been identified</li></ul>



# Case Study: Establish Migration Context <sub>1</sub>

---

DoD organization tasked with developing services that can be used by mission planning and execution applications

MSS is a system for comparison of planned mission against current state to determine if corrective actions should be taken

- In final stages of development

## Drivers

- Migration to services was already a longer-term goal for MSS
- Make developed services available to all mission planning and execution systems

Requirement to demonstrate the feasibility of one component as a service being used by one mission planning and execution system within 6 months and to migrate the full system to services in two years





# Case Study: Establish Migration Context <sub>2</sub>

---

Standard Web Services environment is target SOA environment

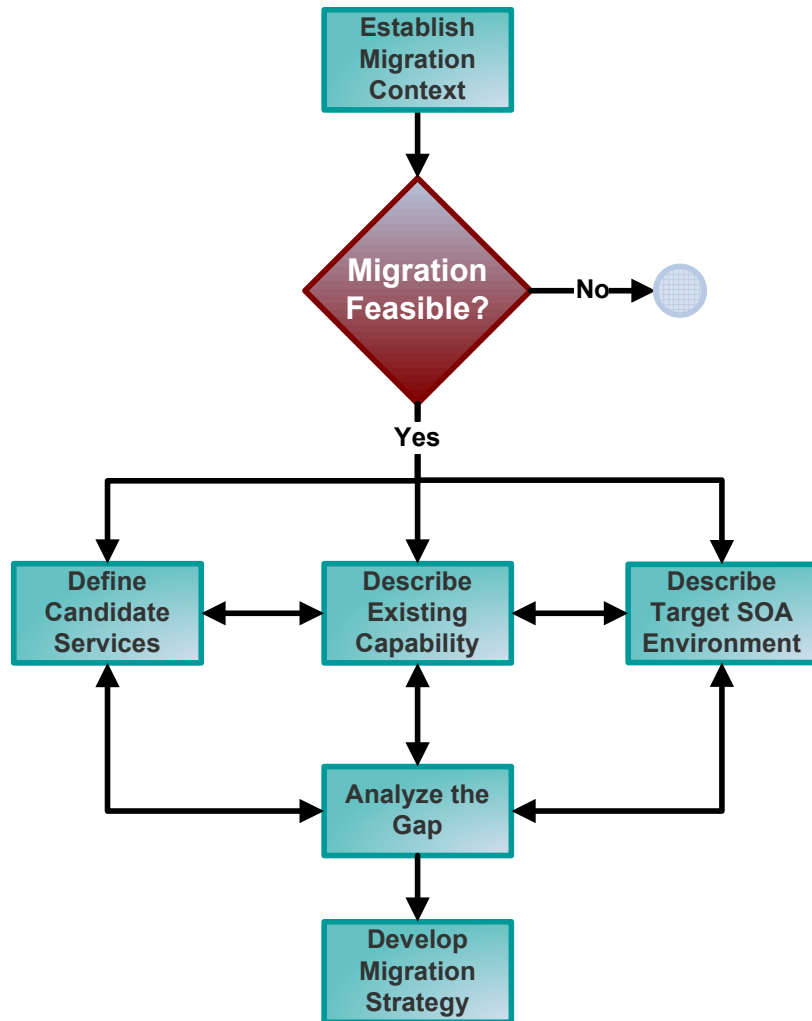
- Not clear that this will be the future environment for the developed services

Representatives from the legacy system and a representative from a mission planning and execution application (service consumer) agreed on the following candidate services

- *AvailablePlans*: Provides list of available plans that are being reasoned about.
- *TrackedTasksPerPlan*: Provides list of tasks that are being tracked for a certain plan.
- *TaskStatus*: Provides the status for a given task in a given plan.
- *SetTaskAlert*: Alerts when a given task in a given plan satisfies a certain condition



# Checkpoint for Migration Feasibility



Decision to continue with the process has to be made

Potential outcomes at this point are

- The migration is initially feasible
- The migration has potential but requires additional information to make an informed decision
- The migration is not feasible



# Case Study: Migration Feasibility

---

## Decision: Migration feasible

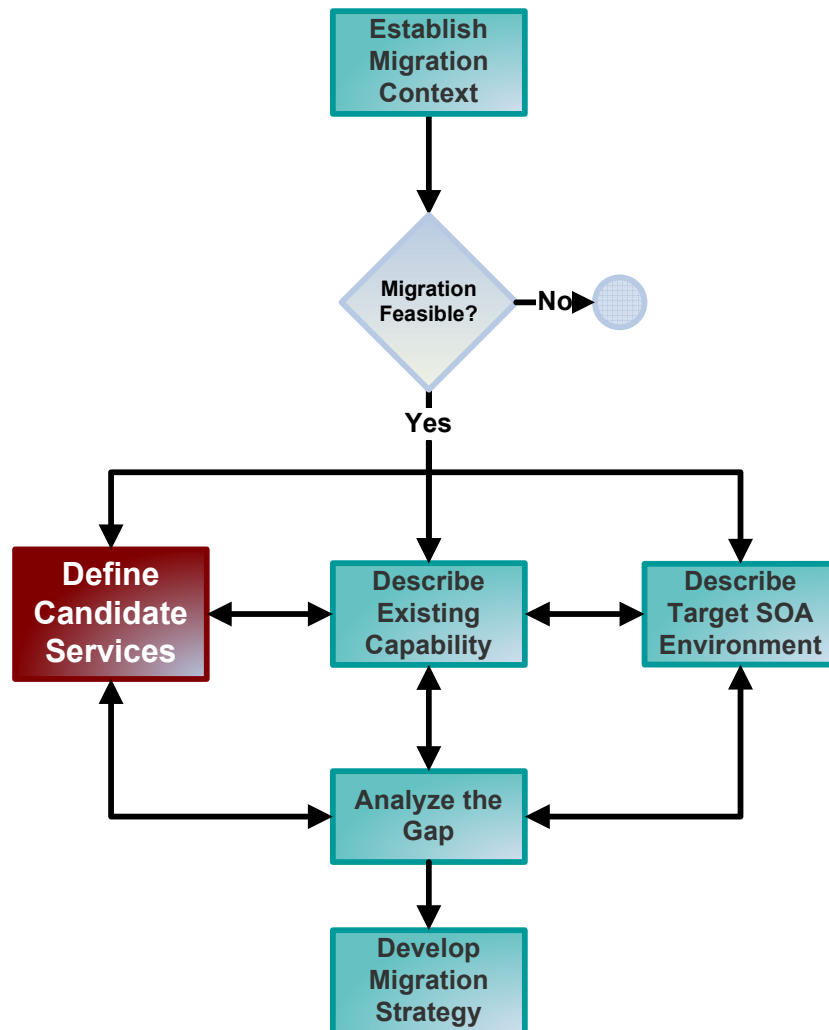
- Availability of stakeholders from the service provider and a service consumer
- Good understanding of the legacy system
- Request-response nature of the identified services
- Reasonable initial mapping of services to components

## Migration issues identified in this activity

- Short-term goal for the migration is different from long-term goal migration
  - Work to accomplish the short-term goal might have to be redone in order to accomplish the long-term goal
- System is a single-user, single-plan system
  - When capabilities are migrated to services, it will have to support multiple users and multiple plans



# Define Candidate Services



Select a small number of services, usually 3-4, from the initial list of candidate services

For these candidate services, the end goal is to fully specify inputs and outputs



# Case Study: Define Candidate Services

---

The list of services identified in the previous step was considered reasonable for analysis

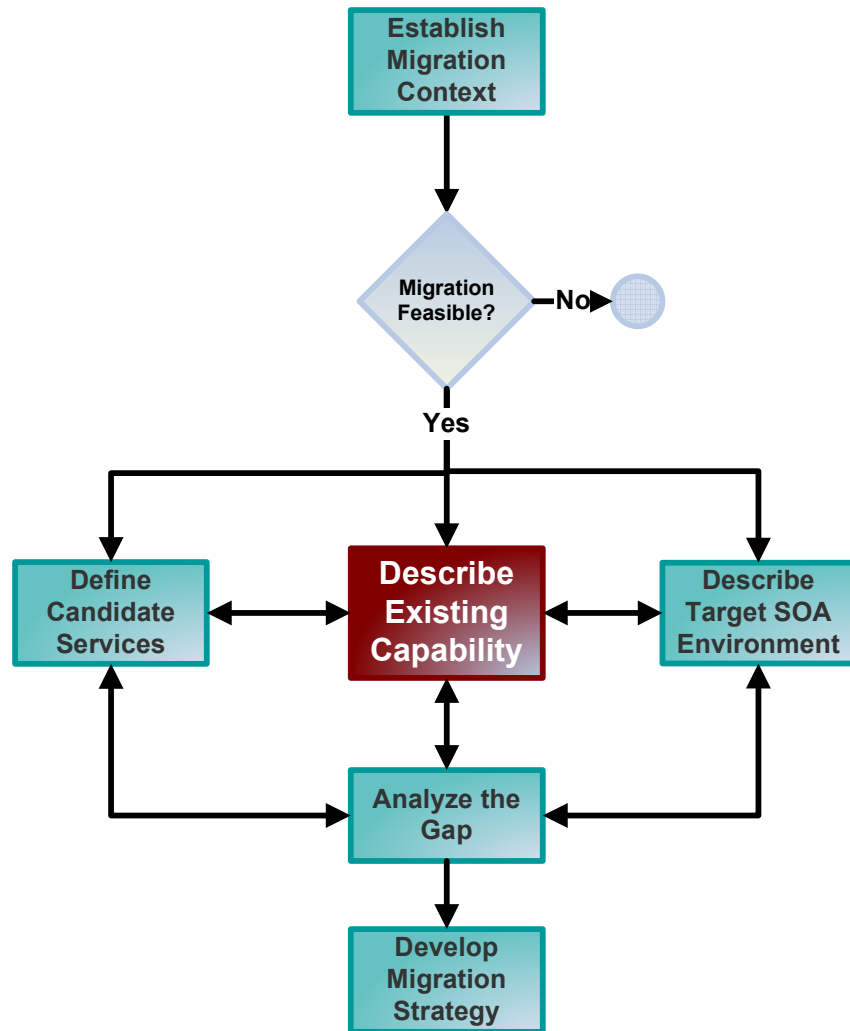
Inputs and outputs were next identified in detail for each of these services

Migration issues identified in this activity

- *SetTaskAlert* requires (1) alert is set up to respond to certain conditions and (2) service consumer is alerted when the condition is reached
  - Handling of events in service-oriented environments is new—SOA 2.0
- Unclear how the alert mechanism is going to be implemented
  - SOA infrastructure would need to have a way to call back the service consumer
  - There might also be firewall issues
- Complexity of alert conditions is high
  - Service consumer interface will have to replicate this complexity or conditions would have to be made simpler or limited



# Describe Existing Capability



Obtain descriptive data about legacy components

- Name, function, size, language, operating platform, age of legacy components, etc.

Question technical personnel about

- Architecture and design paradigms
- Complexity, coupling, interfaces
- Quality of documentation
- Component/product dependencies

Gather data about

- Quality, maturity, existing problems
- Change history
- User satisfaction



# Describe Existing Capability: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
<b>Legacy System Characteristics</b>	<ul style="list-style-type: none"> <li>• What is the history of the system?</li> <li>• Is the system a proof of concept, prototype, under development, in testing, or a fielded system?</li> <li>• What system documentation is available?</li> <li>• Does the system have interfaces to other systems?</li> <li>• What are potential locking, persistence, or transaction problems if accessed by multiple users when migrated to services?</li> </ul>	<ul style="list-style-type: none"> <li>• Planned development concurrent with service migration</li> <li>• Limited system documentation</li> <li>• Interfaces to other systems will open doors to service consumers</li> <li>• Single-user system may have problems in a multi-user environment</li> </ul>
<b>Legacy System Architecture</b>	<ul style="list-style-type: none"> <li>• What architecture views are available?</li> <li>• What are the major modules of the system and dependencies between modules?</li> <li>• Is user interface code separate from the business logic code?</li> <li>• Are there any design paradigms or patterns implemented in the system?</li> <li>• What are the key quality attributes built into the current architecture of the system?</li> </ul>	<ul style="list-style-type: none"> <li>• Lack of architecture documentation may lead to underestimation of complexity</li> <li>• Tight coupling between user interface code and business logic code increases effort</li> <li>• Undocumented violations of design patterns may cause problems</li> <li>• Key quality attributes may not hold true in a services environment</li> </ul>
<b>Code Characteristics</b>	<ul style="list-style-type: none"> <li>• What code documentation is available?</li> <li>• What coding standards are followed?</li> </ul>	<ul style="list-style-type: none"> <li>• Poor coding practices will increase migration effort</li> </ul>



# Case Study: Describe Existing Capability

---

## MSS characteristics

- In demonstration state
- Written in C++, C# and Managed C++ in a Visual Studio 2005 development environment
- Runs on a Windows XP platform
- Size of the full system is approximately 13,000 lines of code
- Code documentation was rated between *Fair* and *Good* by its developers

Several architecture views were presented that were useful for understanding the system

MSS relies on an external planning system (PS) for plan data and situational awareness data

- PS is being targeted for migration to services in the future

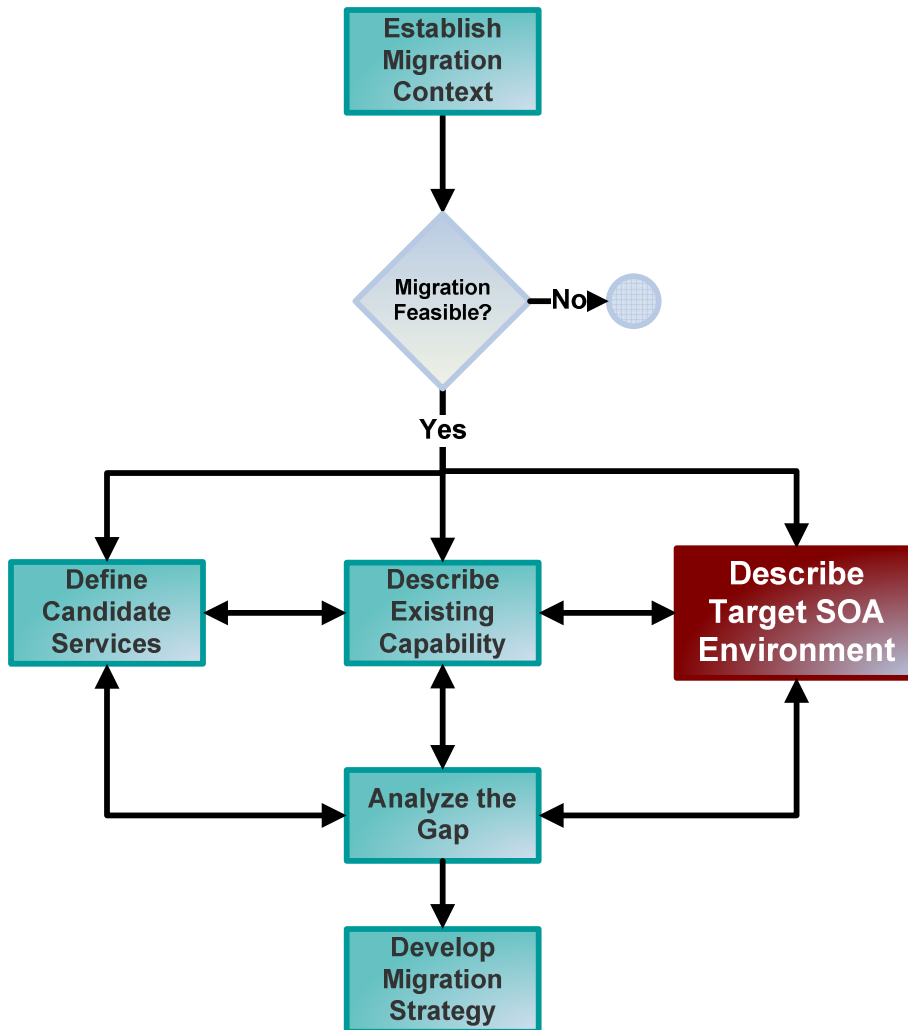
## Migration issues identified in this activity

- Documentation for most of the analyzed classes was determined *Fair*
  - Could be an issue if original developers do not perform the migration
- Currently a large amount of communication between MSS and PS
  - Unclear how performance will be affected when this communication takes place using services (they currently reside on the same machine)
- Task alert functionality is not currently implemented in MSS
  - Still unknowns about the specifics of the implementation





# Describe Target SOA Environment



- Identify the impact of specific technologies, standards, and guidelines for service implementation
- Determine state of target SOA environment
- Identify how services would interact with the SOA environment
- Determine QoS expectations and execution environment for services

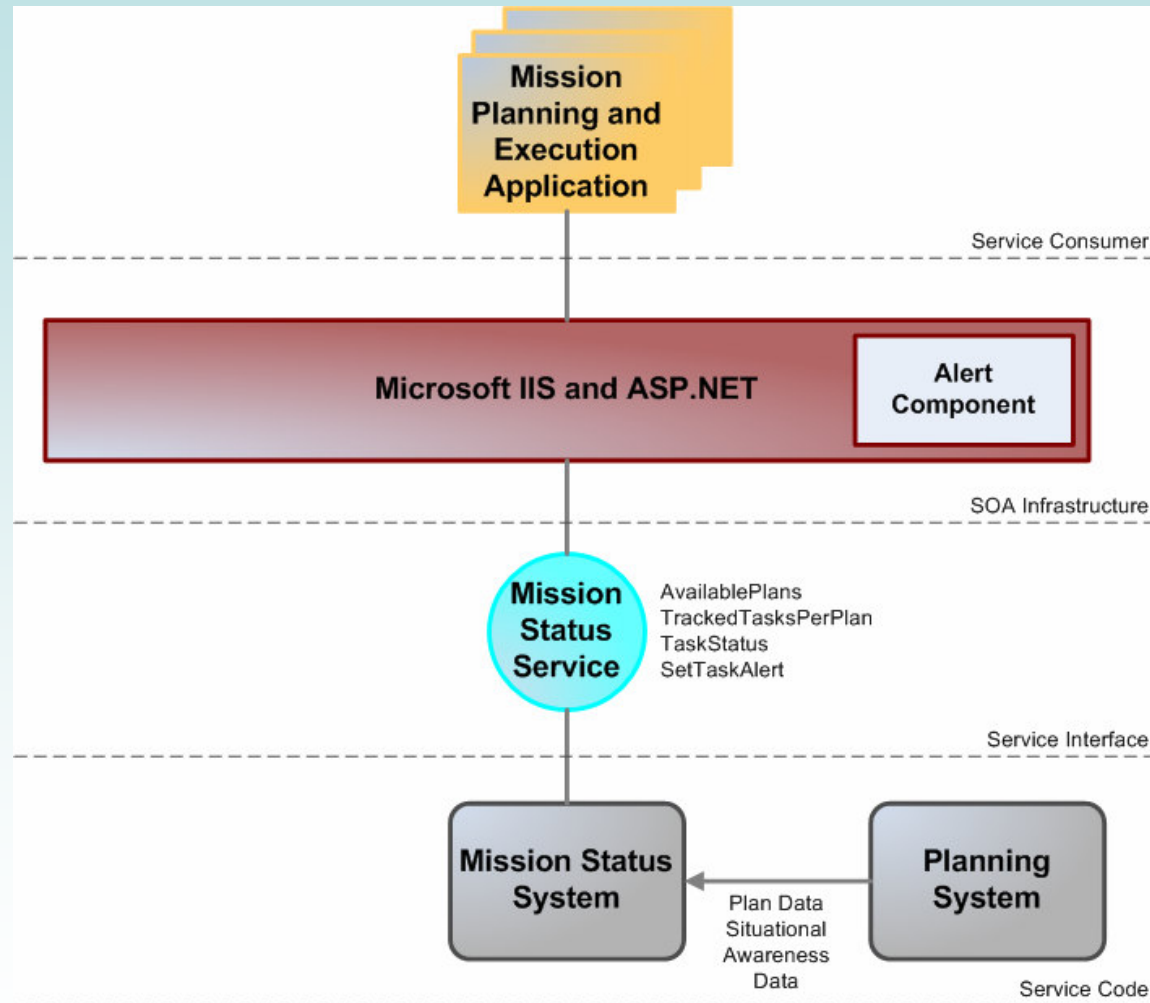


# Describe Target SOA Environment: SMIG Examples

Discussion Topic	Related Questions	Potential Migration Issues
<b>SOA Environment Characteristics</b>	<ul style="list-style-type: none"> <li>• What is the status of the Target SOA environment?</li> <li>• What are the major components of the SOA infrastructure?</li> <li>• Does the target SOA environment provide infrastructure services; i.e. communication, discovery, security, data storage?</li> <li>• What is the communication model?</li> <li>• What constraints does the target SOA environment impose on services?</li> <li>• Does the legacy system have any behavior that would be incompatible with the target SOA environment?</li> <li>• Once developed, where will services execute?</li> </ul>	<ul style="list-style-type: none"> <li>• Target SOA environment largely undefined</li> <li>• Redundancy/conflicts between infrastructure services and legacy code</li> <li>• Lack of tools to support legacy code migration to target infrastructure</li> <li>• Compliance with constraints requires major effort</li> <li>• Architectural mismatch</li> <li>• No thought given to service deployment and execution</li> </ul>
<b>Support</b>	<ul style="list-style-type: none"> <li>• Do you have to provide automated test scripts for the services and make them publicly available?</li> <li>• How will service consumers report problems and provide feedback?</li> <li>• How will service consumers be informed of potential changes in service interfaces and down time due to upgrades or problems?</li> </ul>	<ul style="list-style-type: none"> <li>• Underestimation of effort to provide service consumer support</li> <li>• Lack of awareness of support requirements</li> </ul>



# Case Study: Notional SOA-Based System Architecture



# Case Study: Describe Target SOA Environment

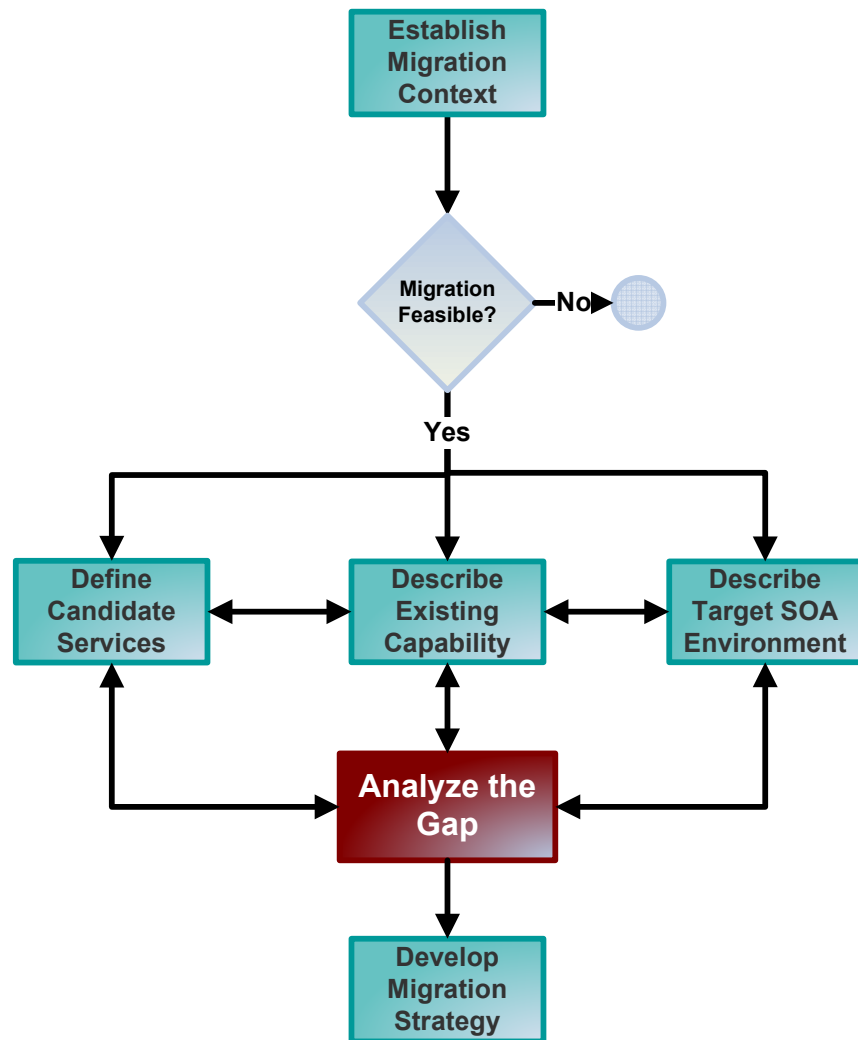
---

## Migration issues identified in this activity

- Not known if the publish-subscribe component will allow someone to subscribe on behalf of a third party
  - If it does not, the service consumer will have to be aware of the dependency on the publish-subscribe component
  - Ideal situation would be for the *SetTaskAlert* service code to subscribe on behalf of the service consumer, so that the service consumer is not affected if the alert mechanism changes
- If the service consumer has to be set up as a Web server, it would have to be configured so that it accepts incoming messages from the publish-subscribe component
  - Potential security concern



# Analyze the Gap



- Define effort, risk and cost to convert legacy components into services, given candidate service requirements and target SOA characteristics
- Determine need for additional analyses



# Case Study: Analyze the Gap

---

Developers were asked to

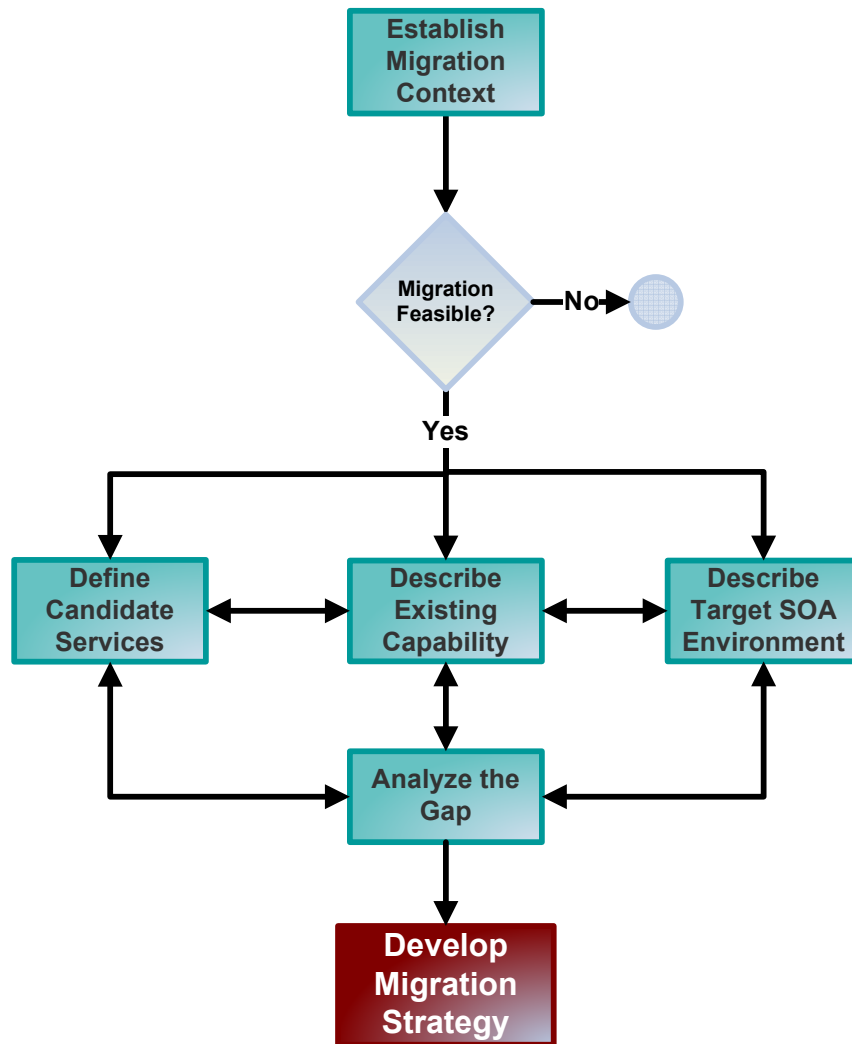
- Describe the details of the changes that would have to be made to the code given the service requirements, the service inputs and outputs, as well as the characteristics and components of the target SOA environment
- Provide an estimate of the effort required to make these changes

No code analysis or architecture reconstruction was necessary because

- Original developers were involved in the process
- Input was credible
- Architecture documentation and knowledge of the system were acceptable



# Develop Migration Strategy



Develop one or more migration strategies that may include

- Order in which to create services
- Guidelines for creation of services
  - Service reference architectures
  - Source of service code (legacy, COTS, external services, etc.)
  - Mechanism—wrapping, rewriting, extraction, new
- Specific migration paths to follow (e.g. wrap first and rewrite later)
- Needs for training, technology evaluation, market research, etc.



# Case Study: Migration Strategy <sub>1</sub>

---

1. Define scope of initial migration for short-term feasibility demonstration
  - Decision of what services to implement and whether they would have time to separate MSS from PS
2. Define scope of subsequent iterations
  - Will depend on additional services to be created from MSS as well as progress made in the migration of PS to services
3. Finalize service inputs and outputs
  - Alert condition structure was still undefined
4. Gather information about the publish-subscribe component to be used as the mechanism for alert capability

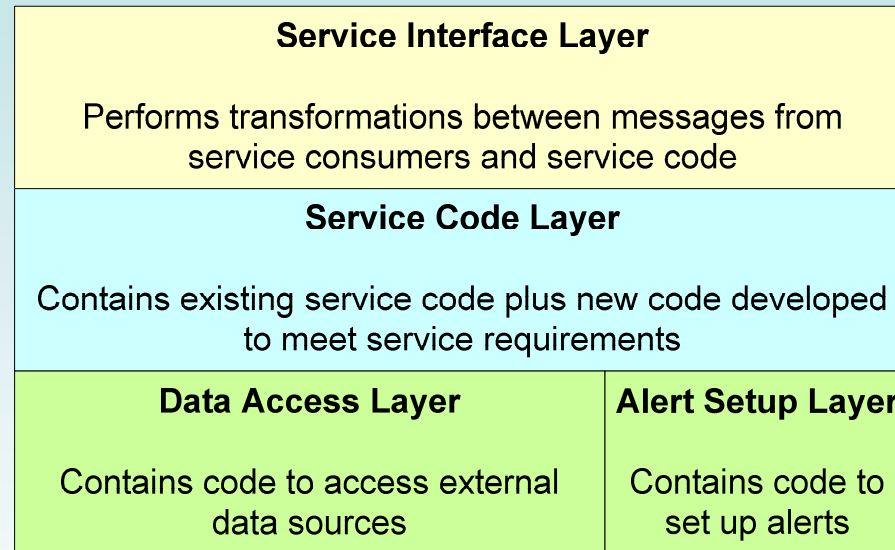




# Case Study: Migration Strategy <sub>2</sub>

---

## 5. Create a service reference architecture



## 6. Adjust estimates

## 7. Create MSS services using the service reference architecture

## 8. Document lessons learned



# Agenda

---

SOA Basics

SMART (Service Migration and Reuse Technique)

**Conclusions** 



# Conclusions

---

SOA offers significant potential for leveraging investments in legacy systems by providing a modern interface to existing capabilities, as well as exposing legacy functionality to a greater number of users

SMART analyzes the viability of reusing legacy components as the basis for services by answering these questions:

- Does it make sense to migrate the legacy system to services?
- What services make sense to develop?
- What components can be mined to derive these services?
- What changes are needed to accomplish the migration?
- What migration strategies are most appropriate?
- What are the preliminary estimates of cost and risk?

